PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-083310
(43)Date of publication of application : 31.03.1998

(51)Int.Cl. G06F 9/445
G06F 9/06
G06F 12/14

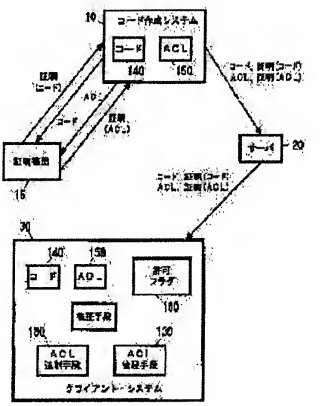
(21)Application number : 09-151768
(22)Date of filing : 10.06.1997

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : DAN ASIT
RAMASWAMI RAJIV
SITARAM DINKAR

(30)Priority
Priority number : 96 661517 Priority date : 11.06.1996 Priority country : US

(54) PROGRAM CODE DISTRIBUTING METHOD AND ITS SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide an authentification system for allowing a relied third party to confirm the author of a program and to sign a certification for guaranteeing the perfection of the program.
SOLUTION: A program code 140 is capsuled together with a guarantee and access control list(ACL) 150. ACL 150 describes an allowable condition and a resource required by the code 140. A forcing mechanism assigns the allowanble condition of a system and a resource according to ACL 150. For example, a code preparing system 10 communicates with a certifying organization 15 being the relied third party. The organization 15 issues the cerificate of the code 140 and the certificate of ACL 150 of the code 140. Once the certificate is issued, nobody can change the code 140 and ACL 150 without invalidating the certificate. The code 140, its ACL 150 and their certificates are stored in a server.



(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平10－83310

(43)公開日　平成10年(1998) 3月31日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/445			Ｇ０６Ｆ　9/06	４２０Ｊ	
9/06	５５０			５５０Ｚ	
12/14	３１０		12/14	３１０Ｋ	
				３１０Ｚ	

審査請求　未請求　請求項の数18　ＯＬ　（全 16 頁）

(21)出願番号　特願平9－151768

(22)出願日　平成９年(1997) ６月10日

(31)優先権主張番号　０８／６６１５１７

(32)優先日　1996年６月11日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　アスィット・ダン
アメリカ合衆国10604、　ニューヨーク州　ウェスト　ハリソン　ゲインスボーグ　アヴェニュー　75

(74)代理人　弁理士　坂口　博　（外１名）

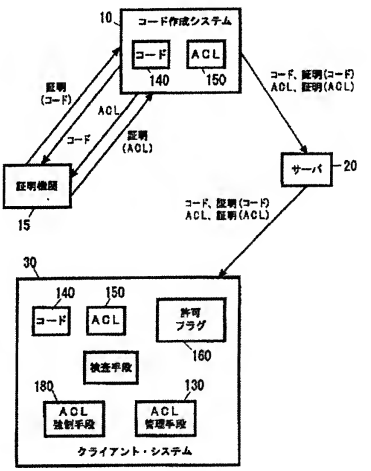
最終頁に続く

(54)【発明の名称】　プログラム・コードの配布方法及びシステム

(57)【要約】

【課題】　信頼される第三者がプログラムの作者を確認しプログラムの完全性を保証するための証明に署名する認証方式を提供すること。

【解決手段】　プログラム・コードは証明およびアクセス制御リスト（ＡＣＬ）と共にカプセル化されている。アクセス制御リストはコードによって要求される許容条件とリソースを記述する。強制メカニズムはアクセス制御リストに従ってシステムの許容条件とリソースを割り振る。実施例において、コード作成システムは、信頼される第三者である証明機関と通信する。証明機関はコードの証明と、そのコードのアクセス制御リストの証明とを発行する。一度、証明が発行されると、いかなる者もその証明を無効にすることなくコードまたはアクセス制御リストを変更することはできない。コード、そのアクセス制御リスト、およびそれらの証明はサーバ中に記憶される。



【特許請求の範囲】
【請求項１】プログラム・コードの配布方法であって、信頼される第三者の証明を受け取りシステムへ提供するステップを含み、上記プログラム・コードの検査された無害の動作に必要なリソースと許容条件を、上記信頼される第三者の証明の中にコンピュータで読み取り可能に記述したことを特徴とする、プログラム・コードの配布方法。
【請求項２】請求項１において、上記証明をプログラム・コードと共にカプセル化するステップを含む、プログラム・コードの配布方法。
【請求項３】請求項１において、上記受け取りシステムが上記証明を読み取るステップと、上記証明中に指定された許容条件を超過しないように上記受け取りシステムのリソースと許容条件を割り振るステップとを含む、プログラム・コードの配布方法。
【請求項４】請求項２において、上記証明に関連してユーザが選択したオプションに従って、上記受け取りシステムのリソースに対する上記プログラム・コードのアクセスと許容条件を否定または肯定するステップを含む、プログラム・コードの配布方法。
【請求項５】請求項１において、コンピュータで読み取り可能に記述した上記リソースと許容条件が暗号形式で上記受け取りシステムへ提供されることを特徴とする、プログラム・コードの配布方法。
【請求項６】請求項１において、暗号化された検査データが上記証明の中に含まれており、上記受け取りシステムが上記検査データを解読して上記リソースと許容条件の記述の完全性を検査することを特徴とする、プログラム・コードの配布方法。
【請求項７】請求項３において、必要なリソースの記述の中に、上記プログラム・コードによって使用される各リソースの量と、そのリソースの最大許容消費率の両方が記述されていることを特徴とする、プログラム・コードの配布方法。
【請求項８】請求項３において、必要な許容条件の記述の中に、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。
【請求項９】請求項１において、上記プログラム・コードの機能性が、上記第三者による検査に従って上記第三者の証明の中に記述されていることを特徴とする、プログラム・コードの配布方法。
【請求項１０】請求項１において、上記プログラム・コードが、サーバからプログラム・オブジェクトとしてダウンロードされるアプレットであることを特徴とする、プログラム・コードの配布方法。
【請求項１１】請求項１において、上記受け取りシステムの中で上記プログラム・コードの異なったユーザに対しては異なったリソースと許容条件を割り振ることを特徴とする、プログラム・コードの配布方法。
【請求項１２】請求項１１において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードのインストール時に決定されることを特徴とする、プログラム・コードの配布方法。
【請求項１３】請求項１１において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードの実行時に決定されることを特徴とする、プログラム・コードの配布方法。
【請求項１４】プログラム・コードの配布方法であって、信頼される第三者によるプログラム・コードの証明の中に、該プログラム・コードの検査された無害の動作に必要なリソースと許容条件をコンピュータで読み取り可能に記述したものを入れ、その証明を上記プログラム・コードと共にカプセル化して暗号形式で受け取りシステムへ提供するステップと、上記受け取りシステムが上記暗号形式の証明を読み取るステップと、上記読み取られた証明の完全性を上記受け取りシステムによって決定するステップと、上記完全性が検査された後に、上記証明の中で指定された許容条件を超過しないようにユーザが選択したオプションに従って上記受け取りシステムのリソースと許容条件を割り振るステップと、上記割り振りに従って上記プログラム・コードを実行するステップとを含み、上記必要なリソースの記述の中には、上記プログラム・コードによって使用される各リソースの量とその最大許容消費率との両方が記述され、上記許容条件の記述の中には、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。
【請求項１５】コンピュータ・システムへプログラムとデータを移入する移入装置と、上記コンピュータ・システムの動作を制御するオペレーティング・システムと、コードの検査された無害の動作に必要なリソースをコンピュータで読み取り可能に記述したものを、上記データから抽出してそれを所与のプログラムに関連づけるアクセス・ロジックと、該アクセス・ロジックに含まれて、上記コンピュータで読み取り可能に記述したものの完全性を示す検査データを生成する完全性検査ロジックと、上記オペレーティング・システムに接続され、上記検査データに応答して多数のリソースの各々について上記受け取りシステムの中で消費量と消費率を追跡し割り振って、上記記述の中に指定された割り振りを超過しないようにする強制ロジックと、上記割り振りに従ってプログラム・コードを実行するプロセッサと、を具備するコンピュータ・システム。
【請求項１６】請求項１５において、ランダム・アクセス・メモリに記憶されたデータ構造が上記強制ロジックに接続されており、該データ構造の中には、消費された実際のリソースを追跡する第１のフィールドと、リソー

スの消費率を追跡する第２のフィールドと、上記記述から引き出されたリソース消費の限度を記憶する第３のフィールドと、上記記述から引き出されたリソース消費率の限度を記憶する第４のフィールドとが含まれていることを特徴とする、コンピュータ・システム。

【請求項１７】請求項１５において、上記アクセス・ロジックの中に、プログラム・コードを含むパッケージから上記記述を非カプセル化する手段と該記述を解読する手段とが含まれていることを特徴とする、コンピュータ・システム。

【請求項１８】請求項１５において、上記強制ロジックの中に、上記記述に関連してユーザが選択したオプションに従ってコンピュータ・システムのリソースに対するコード・アクセスと許容条件を否定または肯定する手段が含まれていることを特徴とする、コンピュータ・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般的には、ネットワークのような頒布システムを使用してソフトウェアを送付することに関する。

【０００２】

【従来の技術】ソフトウェアの頒布業は最近の大きな産業である。今日、ソフトウェアはディスケットおよびＣＤ－ＲＯＭを介して頒布され、また次第にネットワークを介して頒布されるようになった。さらに、最近では、単にインターネットを介して遠隔サイトからコードをダウンロードするだけでなく、クライアントはサーバからアプレットをダウンロードして実行することができる。これは、ジャバ（Ｊａｖａ）・プログラミング言語、またはテレスクリプトのような他の言語で提案されているパラダイムである。

【０００３】他から得られたコードに伴う重要な問題は、セキュリティの問題である。たとえば、ＵＮＩＸオペレーティング・システムでは、コードはクライアントのすべての特権を具備しているクライアント・シェルの中で実行される。そのような特権には、クライアントのすべてのファイルにアクセスすること、メールを送ること、さらには不法侵入を試みることも含まれる。ジャバは、このようなアプレットの問題を解決するために、非常に制限された環境でアプレットを実行する。その結果、ジャバのアプレットの有用性と機能性は制限される。他方、ジャバのアプリケーションはオペレーティング・システムによって提供されるセキュリティに依存し、どのような認証（ａｕｔｈｅｎｔｉｃａｔｉｏｎ）にもなじまない。したがって、それらアプリケーションは他のコードと同じようなセキュリティの問題を提起する。

【０００４】認証の１つの形式は、「Ｔｒｕｓｔｅｄ Ｄｉｓｔｒｉｂｕｔｉｏｎ ｏｆＳｏｆｔｗａｒｅ Ｏｖｅｒ ｔｈｅ Ｉｎｔｅｒｎｅｔ」（Ａ．Ｄ．Ｒｕｂｉｎ，Ｉｎｔｅｒｎｅｔ Ｓｏｃｉｅｔｙ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｎｅｔｗｏｒｋ ａｎｄ Ｄｉｓｔｒｉｂｕｔｅｄ Ｓｅｃｕｒｉｔｙ，１９９５）に提案されている。そこでは、信頼される第三者が証明に署名して、プログラムの作者を確認するとともにプログラムの完全性を保証している。

【０００５】

【発明が解決しようとする課題】この方法によって、クライアントはコードの認証を検証できるが、この方法は、コードに関連した許容条件を柔軟に指定するものではなく、またクライアントがこれらの許容条件を自動的に強制する方法を提供するものでもない。

【０００６】

【課題を解決するための手段】本発明は、信頼される第三者が証明に署名して、プログラムの作者を確認するとともにプログラムの完全性を保証するものであるが、プログラム・コードはカプセル化されている（すなわち、証明およびアクセス制御リストと関連づけられている）。アクセス制御リストはコードによって要求される許容条件とリソースを記述している。さらに、本発明は、アクセス制御リストに従ってシステムに許容条件とリソースを割り振る強制メカニズムを提供する。

【０００７】実施例において、コード作成システムは、信頼される第三者である証明機関と通信する。証明機関はコードの証明と、そのコードに関するアクセス制御リストの証明を発行する。一度、証明が発行されると、いかなる者も、その証明を無効にすることなしに、そのコードまたはアクセス制御リストを変更することはできない。コードおよびそのアクセス制御リストは、それらの証明と共にサーバに記憶される。コードまたはアクセス制御リストをダウンロードするクライアントは、コードないしアクセス制御リストの完全性を検査することができ、システムは許容条件とリソースを超過しないようにアクセス制御リストを実施することができる。

【０００８】

【発明の実施の形態】図１は、本発明に従うコード送付および証明システムのブロック図である。そこには１つまたは複数のコード作成システム１０、証明機関１５、１つまたは複数のサーバ２０、および１つまたは複数のクライアント・システム３０が示される。クライアント・システムは通常の広域ネットワーク（ＷＡＮ）またはローカル・エリア・ネットワーク（ＬＡＮ）を介してサーバへ接続される。コード作成システムは、証明機関によって証明してもらいたいプログラム・コード１４０と、そのコードに関するアクセス制御リスト（ＡＣＬ）１５０とを含む。証明機関は、クライアント・システムによって広く使用され知られている公用キーＫを提供する。さらに、証明機関は、それだけが知っている私用キーＰを保持している。コードの証明を提供するために

は、証明機関はコード名とコードの暗号ハッシュ（ｈａｓｈ）を含む証明を作成し、私用キーを使用してそれに署名する。今や、その証明は証明機関の署名を無効にすることなしに変更することはできない。同様に、証明機関は、コードに関連したアクセス制御リスト用の証明にも署名する（もし望むなら、コードとそのアクセス制御リストの双方のために、ただ１つの証明にサインするようにしてもよい）。図１の実施例では、クライアント・システムは、ＷＡＮまたはＬＡＮを介して証明、アクセス制御リスト、およびコードを受け取る。しかし、その方法に替えて、クライアント・システムは、取り外し可能記憶媒体を介して、証明されたコードを受け取ることができる点に注意されたい。このような記憶媒体は、フロッピ・ドライブまたは光学ディスク・ドライブのようなローカル移入装置によって読み取られる。

【０００９】図２に示されたクライアント・システムは検査手段１１０、アクセス制御リスト（ＡＣＬ）管理手段１３０、実行手段１７０、アクセス制御リスト（ＡＣＬ）強制手段１８０、およびクライアント・インタフェース１９０を含む。さらに、クライアント・システムは通常のＣＰＵおよび関連制御ロジック１６２、通信コントローラ／ネットワーク・インタフェース１９４、ＣＤ－ＲＯＭ／フロッピ・ディスク・サブシステム１９６、および他のリソース（たとえば、ネットワーク・アクセス・モジュール、ディスプレイ・サブシステム、および各種の特殊目的アダプタ）を含む。当業者にとって、クライアント・システムが他の多くの通常のコンポーネントを含むことは明らかであろうが、それらについては、ここで詳細に説明しない。

【００１０】アクセス制御リスト管理手段１３０とアクセス制御リスト強制手段１８０は、プログラム・コードとして実施するのが望ましい。その動作については後で詳細に説明する。実行手段はクライアントのオペレーティング・システム（図示されず）の通常の部分であり、クライアント・システム上で、移入されたコードを実行するために使用される。クライアント・インタフェース１９０は画面グラフィカル・ユーザ・インタフェースのフロントとして実施することができ、アクセス制御リスト管理手段１３０とクライアントとの連絡を可能にする。（たとえば、それによって、クライアントは、指定されたリソースへのプログラム・アクセスを許すか許さないかをアクセス制御リスト管理手段１３０に命令したり、プログラム・アクセスを管理したりすることができる。）検査手段１１０はプログラム・コードとして実施することが望ましい。それは解読モジュールを含み、アクセス制御リストと共に移入されたコードの認証を検査する。さらに、検査手段はハッシング（ｈａｓｈｉｎｇ）モジュールを含む。このモジュールは、移入されたコードとアクセス制御リストの完全性をチェックする。

【００１１】「コード／アクセス制御リストおよび証明」１００がダウンロードされると、まず検査手段１１０は証明の上にある証明機関のサインが正しいかどうかを検査する（証明機関の既知の公用キーを使用する）。次に、検査手段はコード／アクセス制御リストの暗号ハッシュを計算して、それが証明中の値と一致するかどうかを検査する。サインが正しくないとき（すなわち、ハッシュが一致しないとき）、そのコードとアクセス制御リストは拒絶される（拒絶ステップ１２０）。検査がＯＫであれば、アクセス制御リスト管理手段１３０が呼び出される。アクセス制御リスト管理手段はクライアント・インタフェース１９０を介してクライアントへアクセス制御リスト（後で説明する）を表示し、クライアントがアクセス制御リスト内の個々の項目を許可するか許可しないかを確かめる。アクセス制御リスト管理手段１３０は、クライアント・インタフェースを介してクライアントから命ぜられたとおりに、コード１４０を記憶し、許可フラグ１６０と共にアクセス制御リスト１５０を記憶する。

【００１２】アクセス制御リスト強制手段１８０によってアクセスを制御することのできるリソースには、ファイル・システム、特定のファイル、およびシステム呼び出しのような論理リソースと、ディスク・スペース、ディスク・アクセス、メイン・メモリの割り振り、および各種のシステム・コントローラやアダプタへのアクセスのような物理リソースがある。論理リソース（クライアント・システムの機能）へのアクセスについては、個々の項目が許可されるか許可されないかを、許可フラグ１６０が表示する。物理リソースへのアクセスについては、最大許容量または最大許容消費率を示すために許可フラグを使用できる。

【００１３】さらに、図２はクライアント・システムの異なった部分がどのように相互に動作するかを示す。複数ユーザ・クライアント・システムでは、各ユーザがそれぞれの許可フラグ群を持つことができる。さらに、アクセス制御リストの中には環境変数を入れることができる。実行中に環境変数を変えることによって、個々のユーザはアクセス特権をカスタマイズすることができる。アクセス制御リストと許可フラグはセキュリティ領域に記憶され、この領域の読み出しと更新はアクセス制御リスト強制手段１８０に許される特権である。

【００１４】アクセス制御リスト管理手段は、アクセス制御リストの許可条件を表示または変更するためにクライアントによって任意の時点で呼び出されてよい。コードは実行手段１７０によって実行される。リソースへのアクセスを許す前に、実行手段はアクセス制御リスト強制手段を呼び出して、アクセスの有効性をチェックする。これは、実行手段１７０が、アクセス制御リスト強制手段を呼び出す検査ルーチンへのトラップをコードの中に挿入するという方法によって達成される。アクセス制御リスト強制手段の動作は、後で詳細に説明する。

【００１５】システムでは、必要に応じてコードとそのアクセス制御リストを別個にまたは組み合わせてダウンロードすることができる。たとえば、コード作成システムの方針として、すべてのクライアントへアクセス制御リストを無料で提供するが、コード自体は有料にすることができる。

【００１６】図３はアクセス制御リストを示す。アクセス制御リストは２つの部分から構成される。すなわち、コードによって要求される物理リソースを含む物理リソース・テーブル２００と、コードによって要求される許可条件と論理リソースを含む論理リソース・テーブル２５０である。

【００１７】物理リソース・テーブル２００の中には、各リソースについての行があり、この行の中には物理リソース名２０５、リソース属性２１０、最大許容消費率２１５、および最大許容量２２０がある。リソース属性２１０は、物理装置またはリソース・ディスクに複数の属性があるときに使用される。たとえば、記憶装置についてスペースとＩ／Ｏ数があるような場合である。最大許容消費率２１５と最大許容量２２０は、リソースと属性の最大許容消費率と最大許容消費量である。

【００１８】論理リソース・テーブル２５０は、コードによって要求される外部ルーチン（論理リソースと呼ばれる）への各呼び出しについて１つの行を含む。各行には、論理リソース名２５５とパラメータ・リスト２６０がある。パラメータ・リスト２６０はパラメータ・エントリ２６５のリストを指し示す。各パラメータ・エントリ２６５は、有効なパラメータ範囲の集合を指定する。すなわち、その集合は組み合わせとして有効なパラメータ値の集合である。パラメータ・エントリ２６５の中には、次のパラメータ・エントリを指し示すｎｅｘｔＰＥフィールド２８０も含まれている。各パラメータのパラメータ範囲は２つのフィールドを含む。すなわち、パラメータ・タイプ２７０と、そのパラメータの有効範囲を指定するパラメータ値２７５である。ストリング・パラメータについては、パラメータ・タイプ２７０はＳＴＲであり、パラメータ値２７５はそのストリングについて有効な形式を指定する正規表現のリストである。整数パラメータについては、パラメータ・タイプ２７０はＩＮＴであり、パラメータ値２７５は整数範囲のリストである。パラメータ値２７５の中には環境変数の名前を入れることができ、その場合、実行時にその環境変数の値を置換することが想定されている。

【００１９】クライアントの中のアクセス制御リスト強制手段は、アクセス制御リストの中でコードについて指定された許可条件とリソースが提供され、許可条件／リソースの追加が許されないことを確実にする。

【００２０】アクセス制御リストの強制は静的であっても動的であってもよい。静的強制では、コードが実行される前に全面的に強制が行われ、コードが実行されている間の強制は必要とされない。動的強制では、コードが実行されている間に強制を行わねばならない。証明機関自体がアクセス制御リストを検査し、コードによる超過が生じないことが保証されれば、アクセス制御リストの強制機能はクライアント・システムで不必要となろう。

【００２１】図４は、アクセス制御リスト強制手段１８０によって使用されるデータ構造を示す。実行時物理リソース・テーブル３００には、各リソースについて１つの行がある。リソース名３００、リソース属性３１０、最大許容消費率３１５、および最大許容量３２０は、物理リソース・テーブル２００の対応するフィールドのコピーである。実際消費率３２５および実際使用量３３０の各フィールドは、それぞれ実行時におけるコードの実際の消費率と実際の消費量を追跡するために使用される。実行時論理リソース・テーブル３５０には、必要とされる各論理リソースについての行がある。実行時論理リソース・テーブルは論理リソース・テーブルのコピーであり、それに許可フラグが追加されている。許可フラグは、パラメータの有効なそれぞれの組み合わせについて、その組み合わせがアクセス制御リスト管理手段によって許可されているか許可されていないかを示す。論理リソース名３５５は、論理リソース・テーブル２５０の対応するフィールドのコピーであり、パラメータ・リスト３６０は実行時パラメータ・エントリ３６５のリストを指し示す。実行時パラメータ・エントリのパラメータ・タイプ３７０およびパラメータ値３７５は、論理リソース・テーブル内の対応するフィールドのコピーである。ｎｅｘｔＰＥフィールド３８０は次の実行時パラメータ・エントリ３６５を指し示し、許可フラグ３８５は、この実行時パラメータ・エントリ３６５が許可されているか許可されていないかを示すイエスまたはノーに設定されている。さらに、アクセス制御リスト強制手段はコード開始時点３９５を追跡する。

【００２２】図５は、論理リソースの許可条件がどのようにしてアクセス制御リスト強制手段によって強制されるかを示す。このパスは、コードが外部機能への呼び出しを行うとき常に呼び出される。ステップ４１０で、アクセス制御リスト強制手段１８０はパラメータの数、パラメータの値、および呼び出されつつある機能の名前の位置を決定する。その実際の方法は処理系によって異なる。たとえば、ジャバでは、これらはオペランド・スタック上に置かれている。ステップ４１５では、アクセス制御リスト強制手段は、実行時論理アクセス・テーブル３５０の中で、呼び出されつつある機能の行の位置を決定し、許可フラグがイエスに設定されている最初の実行時パラメータ・エントリ３６５の位置を決定する。機能名が見つからないか、そのような実行時パラメータ・エントリ３６５が存在しない場合、アクセス制御リスト強制手段はステップ４７０へ進み、呼び出しが許されないことを示して終了する。ステップ４２０では、アクセス

制御リスト強制手段は最初のパラメータの値の位置を決定して、それを現在のパラメータにする。ステップ４２５では、アクセス制御リスト強制手段は、現在のパラメータの値が実行時パラメータ・エントリ３６５によって許されているかどうかをチェックする。値が許されていれば、ステップ４３０で、アクセス制御リスト強制手段はもっとパラメータがあるかどうかをチェックする。さらにパラメータがあれば、ステップ４３５で、アクセス制御リスト強制手段は現在のパラメータを次のパラメータに設定しステップ４２５へ戻る。ステップ４３０で、それ以上のパラメータがないと、アクセス制御リスト強制手段はステップ４５０へ進み、呼び出しが許される旨を表示して終了する。

【００２３】ステップ４２５で、許容性のテストが失敗すると、アクセス制御リスト強制手段はステップ４４５へ進む。そこでは、アクセス制御リスト強制手段が実行時論理リソース・テーブル３５０をチェックして、許可フラグ３８５がイエスへ設定されている他の実行時パラメータ・エントリを探す。そのような実行時パラメータ・エントリがなければ、アクセス制御リスト強制手段はステップ４７０へ進み、呼び出しが許されない旨を表示して終了する。そのような実行時パラメータ・エントリ３６５があれば、アクセス制御リスト強制手段はステップ４２０へ進む。

【００２４】図６は、物理要件がどのようにしてアクセス制御リスト強制手段によって強制されるかを示す。アクセス制御リスト強制手段は、リソースを割り振る前にステップ５００で呼び出される。２つのパラメータ、すなわち要求されたリソース量（ＲＥＱＡＭＴ）と消費予想時間（ＣＯＭＰＴ）が提供される。ディスク１／Ｏについては、ＲＥＱＡＭＴはディスク１／Ｏの数であり、ＣＯＭＰＴはディスク１／Ｏが完了する予想時間である。ステップ５０５で、アクセス制御リスト強制手段は、このリソースについて実行時物理リソース・テーブル３００内の行を決定し、最大許容量３２０が指定されているかどうか、および実際使用量３３０にＲＥＱＡＭＴを加えたものが最大許容量３２０を超過するかどうかをチェックする。もし超過すれば、それはステップ５２７で消費が許されない旨を表示する。最大許容量３２０を超過しないとき、ステップ５１０で、アクセス制御リスト強制手段はこのリソースの予測消費率を、（実際使用量３３０　＋　ＲＥＱＡＭＴ）／（現時点　－　コード開始時点３９５　＋　ＣＯＭＰＴ）に従って計算する。ステップ５１５で、アクセス制御リスト強制手段は最大許容消費率３１５が指定されているかどうか、および計画された消費率が最大許容消費率３１５よりも大きいかどうかをチェックする。最大許容消費率３１５が指定されていないか、予測消費率がそれより大きくなければ、アクセス制御リスト強制手段は消費が許される旨を表示して終了する。予測消費率が大きければ、アクセス制御リスト強制手段は、ステップ５３０で、この動作を実行するのに必要な遅延を、（実際使用量３００　＋　ＲＥＱＡＭＴ）／最大許容消費率３１５　－　（現時点　－コード開始時点３９５　－　ＣＯＭＰＴ）に従って計算し、計算された遅延時間だけ消費が遅延しなければならない旨を表示するとともに、必要な遅延時間を戻す。

【００２５】リソースが消費された後、ステップ５５０で、アクセス制御リスト強制手段はリソース消費量（ＣＯＮＳＡＭＴ）を指定するパラメータを使用して呼び出される。呼び出されたアクセス制御リスト強制手段は、実際使用率３２５と実際使用量３３０を更新する。

【００２６】さらに、同じコードについて、受け取りシステムで、異なったユーザには異なったリソースと許容条件を割り振ることができる。それは、コードがインストールされる時点では、アクセス制御リスト強制手段により、異なったユーザへ与えられた特権に注目して、コードに許されたリソースと許容条件に特権を組み合わせることによって行うことができる。この場合、リソースと許容条件の集合は、各ユーザごとに別々に記憶されることが必要であろう。コードが実行されているときには、リソースと許容条件はユーザ・ベースで強制されることになろう。他方、コードの実行中にリソースと許容条件を決定する場合は、アクセス制御リスト強制手段により、異なったユーザへ与えられた特権に注目して、コードに許されたリソースと許容条件に特権を組み合わせることによって決定することができる。

【００２７】図７は、クライアント・システムで行われる完全性検査、実行、および強制の初期設定動作を示す疑似コードである。今までに説明した各種のテーブル、リスト、フラグその他のデータ構造はクライアント・システムのメモリ（たとえば、揮発性ランダム・アクセス・メモリ、ディスク、またはこれらの組み合わせ）の中でインスタンス化されることに注意されたい。前述したように、アクセス制御リスト強制手段とアクセス制御リスト管理手段は、プログラム・コードとして実施するのが望ましい。これらのプログラム・コードは、クライアント・システムのオペレーティング・システムへリンクされるか組み込まれる。図８はアクセス制御リスト管理手段の疑似コードを示す。図９はアクセス制御リスト強制手段の疑似コードを示す。

【００２８】実施例に基づいて本発明を説明してきたが、当業者による各種の変更と改善が可能であると思われる。したがって、本発明の実施例は単なる例であって、制限的に解釈されてはならないことを理解すべきである。本発明の範囲が請求範囲によって限定されることはいうまでもない。

【００２９】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）プログラム・コードの配布方法であって、信頼さ

れる第三者の証明を受け取りシステムへ提供するステップを含み、上記プログラム・コードの検査された無害の動作に必要なリソースと許容条件を、上記信頼される第三者の証明の中にコンピュータで読み取り可能に記述したことを特徴とする、プログラム・コードの配布方法。

（２）上記（１）において、上記証明をプログラム・コードと共にカプセル化するステップを含む、プログラム・コードの配布方法。

（３）上記（１）において、上記受け取りシステムが上記証明を読み取るステップと、上記証明中に指定された許容条件を超過しないように上記受け取りシステムのリソースと許容条件を割り振るステップとを含む、プログラム・コードの配布方法。

（４）上記（２）において、上記証明に関連してユーザが選択したオプションに従って、上記受け取りシステムのリソースに対する上記プログラム・コードのアクセスと許容条件を否定または肯定するステップを含む、プログラム・コードの配布方法。

（５）上記（１）において、コンピュータで読み取り可能に記述した上記リソースと許容条件が暗号形式で上記受け取りシステムへ提供されることを特徴とする、プログラム・コードの配布方法。

（６）上記（１）において、暗号化された検査データが上記証明の中に含まれており、上記受け取りシステムが上記検査データを解読して上記リソースと許容条件の記述の完全性を検査することを特徴とする、プログラム・コードの配布方法。

（７）上記（３）において、必要なリソースの記述の中に、上記プログラム・コードによって使用される各リソースの量と、そのリソースの最大許容消費率の両方が記述されていることを特徴とする、プログラム・コードの配布方法。

（８）上記（３）において、必要な許容条件の記述の中に、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

（９）上記（１）において、上記プログラム・コードの機能性が、上記第三者による検査に従って上記第三者の証明の中に記述されていることを特徴とする、プログラム・コードの配布方法。

（１０）上記（１）において、上記プログラム・コードが、サーバからプログラム・オブジェクトとしてダウンロードされるアプレットであることを特徴とする、プログラム・コードの配布方法。

（１１）上記（１）において、上記受け取りシステムの中で上記プログラム・コードの異なったユーザに対しては異なったリソースと許容条件を割り振ることを特徴とする、プログラム・コードの配布方法。

（１２）上記（１１）において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードのインストール時に決定されることを特徴とする、プログラム・コードの配布方法。

（１３）上記（１１）において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードの実行時に決定されることを特徴とする、プログラム・コードの配布方法。

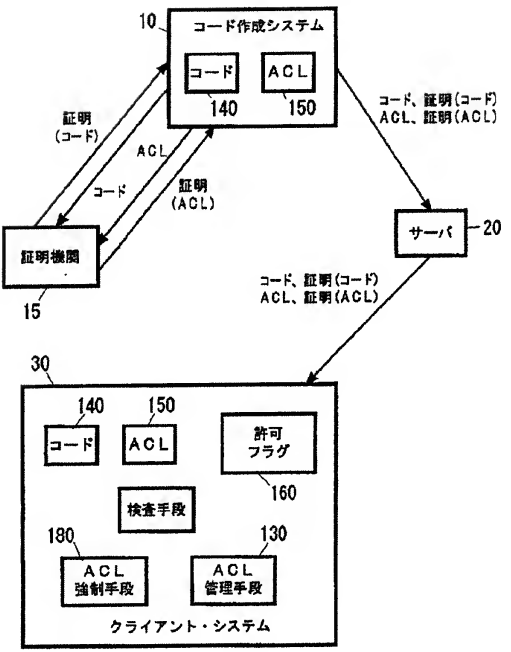
（１４）プログラム・コードの配布方法であって、信頼される第三者によるプログラム・コードの証明の中に、該プログラム・コードの検査された無害の動作に必要なリソースと許容条件をコンピュータで読み取り可能に記述したものを入れ、その証明を上記プログラム・コードと共にカプセル化して暗号形式で受け取りシステムへ提供するステップと、上記受け取りシステムが上記暗号形式の証明を読み取るステップと、上記読み取られた証明の完全性を上記受け取りシステムによって決定するステップと、上記完全性が検査された後に、上記証明の中で指定された許容条件を超過しないようにユーザが選択したオプションに従って上記受け取りシステムのリソースと許容条件を割り振るステップと、上記割り振りに従って上記プログラム・コードを実行するステップとを含み、上記必要なリソースの記述の中には、上記プログラム・コードによって使用される各リソースの量とその最大許容消費率との両方が記述され、上記許容条件の記述の中には、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

（１５）コンピュータ・システムへプログラムとデータを移入する移入装置と、上記コンピュータ・システムの動作を制御するオペレーティング・システムと、コードの検査された無害の動作に必要なリソースをコンピュータで読み取り可能に記述したものを、上記データから抽出してそれを所与のプログラムに関連づけるアクセス・ロジックと、該アクセス・ロジックに含まれて、上記コンピュータで読み取り可能に記述したものの完全性を示す検査データを生成する完全性検査ロジックと、上記オペレーティング・システムに接続され、上記検査データに応答して多数のリソースの各々について上記受け取りシステムの中で消費量と消費率を追跡し割り振って、上記記述の中に指定された割り振りを超過しないようにする強制ロジックと、上記割り振りに従ってプログラム・コードを実行するプロセッサと、を具備するコンピュータ・システム。

（１６）上記（１５）において、ランダム・アクセス・メモリに記憶されたデータ構造が上記強制ロジックに接続されており、該データ構造の中には、消費された実際のリソースを追跡する第１のフィールドと、リソースの消費率を追跡する第２のフィールドと、上記記述から引き出されたリソース消費の限度を記憶する第３のフィールドと、上記記述から引き出されたリソース消費率の限度を記憶する第４のフィールドとが含まれていることを

特徴とする、コンピュータ・システム。
（１７）上記（１５）において、上記アクセス・ロジックの中に、プログラム・コードを含むパッケージから上記記述を非カプセル化する手段と該記述を解読する手段とが含まれていることを特徴とする、コンピュータ・システム。
（１８）上記（１５）において、上記強制ロジックの中に、上記記述に関連してユーザが選択したオプションに従ってコンピュータ・システムのリソースに対するコード・アクセスと許容条件を否定または肯定する手段が含まれていることを特徴とする、コンピュータ・システム。
【図面の簡単な説明】
【図１】本発明に従うコード配布・証明システムを示すブロック図である。
【図２】クライアント・システムの各部分が、実施例で説明される機能を実行するために、相互間でどのように動作するかを示す図である。
【図３】アクセス制御リスト（ＡＣＬ）を示す図である。
【図４】アクセス制御リスト（ＡＣＬ）強制手段によって使用されるデータ構造を示す図である。
【図５】論理リソースの許容条件がアクセス制御リスト（ＡＣＬ）強制手段によってどのように強制されるかを示す図である。
【図６】物理リソースの限度がアクセス制御リスト（ＡＣＬ）強制手段によってどのように強制されるかを示す図である。
【図７】クライアント・システムの完全性検査、実行、および強制の初期設定動作を疑似コードで示した図である。
【図８】アクセス制御リスト（ＡＣＬ）管理手段の疑似コードを示す図である。
【図９】アクセス制御リスト（ＡＣＬ）強制手段の疑似コードを示す図である。
【符号の説明】
１０　コード作成システム
１５　証明機関
２０　サーバ
３０　クライアント・システム
１００　コード／アクセス制御リストおよび証明
１１０　検査手段
１２０　拒絶ステップ
１３０　アクセス制御リスト管理手段
１４０　コード
１５０　アクセス制御リスト
１６０　許可フラグ
１６２　ＣＰＵおよび関連制御ロジック
１７０　実行手段
１８０　アクセス制御リスト強制手段
１９０　クライアント・インタフェース
１９４　通信コントローラ／ネットワーク・インタフェース
１９６　ＣＤ－ＲＯＭ／フロッピ・ディスク・サブシステム
２００　物理リソース・テーブル
２０５　物理リソース名
２１０　リソース属性
２１５　最大許容消費率
２２０　最大許容量
２５０　論理リソース・テーブル
２５５　論理リソース名
２６０　パラメータ・リスト
２６５　パラメータ・エントリ
２７０　パラメータ・タイプ
２７５　パラメータ値
２８０　ｎｅｘｔＰＥ（次のパラメータ・エントリ）フィールド
３００　実行時物理リソース・テーブル
３０５　物理リソース名
３１０　リソース属性
３１５　最大許容消費率
３２０　最大許容量
３２５　実際消費率
３３０　実際使用量
３５０　実行時論理リソース・テーブル
３５５　論理リソース名
３６０　パラメータ・リスト
３６５　実行時パラメータ・エントリ
３７０　パラメータ・タイプ
３７５　パラメータ値
３８０　ｎｅｘｔＰＥ（次のパラメータ・エントリ）フィールド
３８５　許可フラグ
３９５　コード開始時点

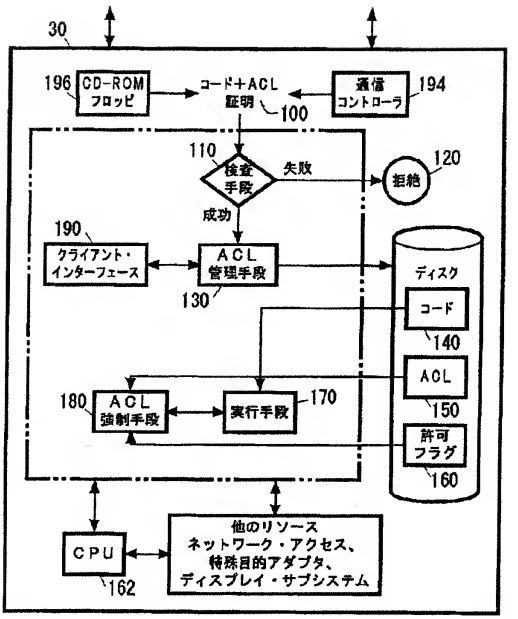
【図１】



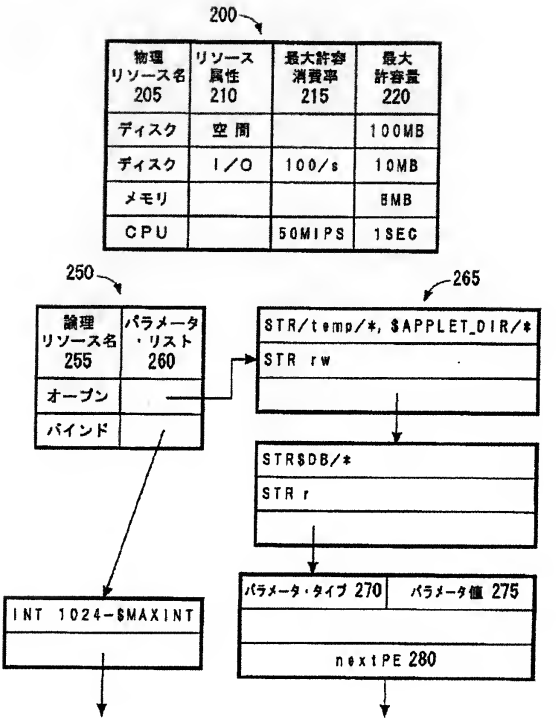
【図８】

```
if 新しく受取られたコードであれば then
        コードとＡＣＬをディスク上に記憶；
プログラムのＡＣＬを表示；
while まだクライアント入力があれば do
      クライアント入力を読取る；
      適切な許可フラグを変更する；
      変更された許可フラグをディスク上に記憶；
end
```

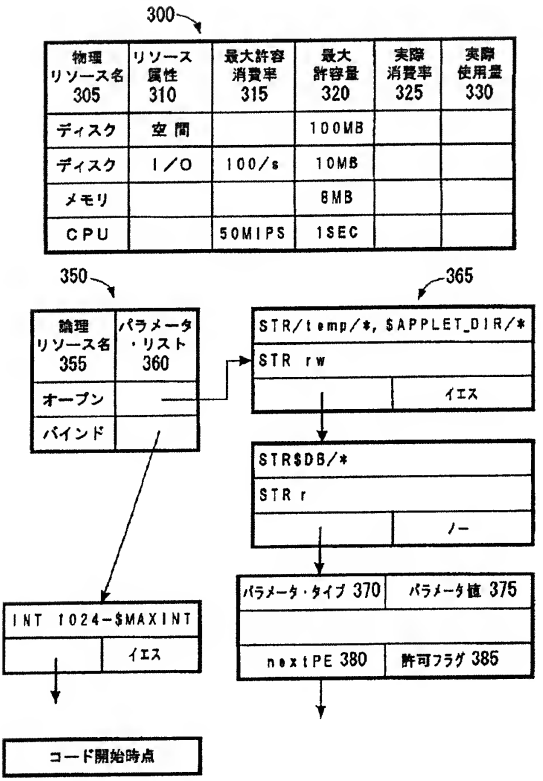
【図２】


30
196
CD-ROM
フロッピ
コード＋ACL
証明
100
通信
コントローラ
194
110
検査
手段
失敗
拒絶
120
成功
190
クライアント・
インターフェース
ACL
管理手段
130
ディスク
コード
140
ACL
150
許可
フラグ
160
180
ACL
強制手段
実行手段
170
CPU
162
他のリソース
ネットワーク・アクセス、
特殊目的アダプタ、
ディスプレイ・サブシステム

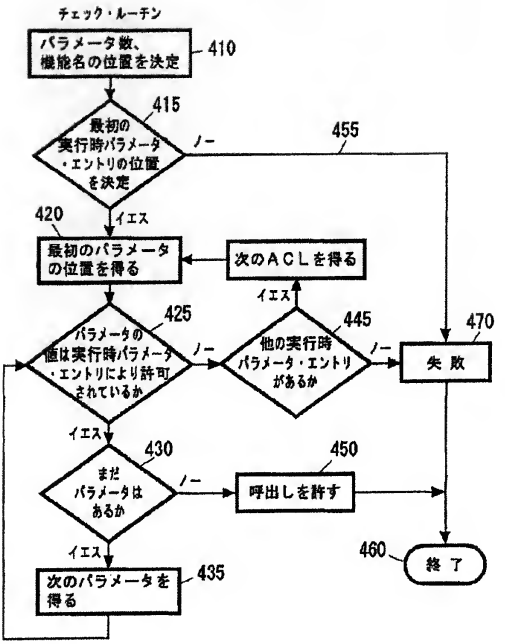
【図3】


200
物理リソース名 205
リソース属性 210
最大許容消費率 215
最大許容量 220
ディスク
空 間
100MB
ディスク
I／O
100/s
10MB
メモリ
8MB
CPU
50MIPS
1SEC
250
論理リソース名 255
パラメータ・リスト 260
オープン
バインド
265
STR/temp/*, $APPLET_DIR/*
STR rw
STR$DB/*
STR r
INT 1024-$MAXINT
パラメータ・タイプ 270
パラメータ値 275
nextPE 280

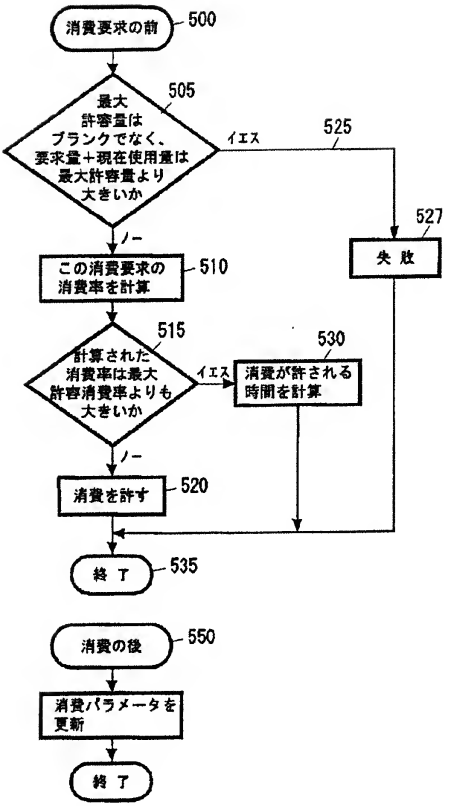
【図４】


300
物理リソース名 305
リソース属性 310
最大許容消費率 315
最大許容量 320
実際消費率 325
実際使用量 330
ディスク
空間
100MB
ディスク
I／O
100/s
10MB
メモリ
8MB
CPU
50MIPS
1SEC
350
論理リソース名 355
パラメータ・リスト 360
オープン
バインド
365
STR/temp/*, $APPLET_DIR/*
STR rw
イエス
STR$DB/*
STR r
ノー
パラメータ・タイプ 370
パラメータ値 375
nextPE 380
許可フラグ 385
INT 1024-$MAXINT
イエス
コード開始時点

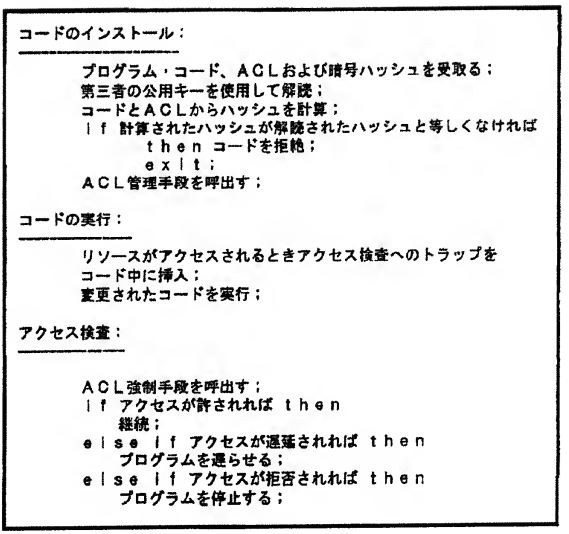
【図５】


チェック・ルーチン
パラメータ数、機能名の位置を決定
410
最初の実行時パラメータ・エントリの位置を決定
415
ノー
455
イエス
420
最初のパラメータの位置を得る
次のＡＣＬを得る
イエス
パラメータの値は実行時パラメータ・エントリにより許可されているか
425
ノー
他の実行時パラメータ・エントリがあるか
445
ノー
470
失敗
イエス
まだパラメータはあるか
430
ノー
呼出しを許す
450
イエス
次のパラメータを得る
435
460
終了

【図６】


消費要求の前
500
最大
許容量は
ブランクでなく、
要求量＋現在使用量は
最大許容量より
大きいか
505
イエス
525
ノー
この消費要求の
消費率を計算
510
失敗
527
計算された
消費率は最大
許容消費率よりも
大きいか
515
イエス
消費が許される
時間を計算
530
ノー
消費を許す
520
終了
535
消費の後
550
消費パラメータを
更新
終了

【図７】


コードのインストール：
プログラム・コード、ＡＣＬおよび暗号ハッシュを受取る；
第三者の公用キーを使用して解読；
コードとＡＣＬからハッシュを計算；
ｉｆ 計算されたハッシュが解読されたハッシュと等しくなければ
ｔｈｅｎ コードを拒絶；
ｅｘｉｔ；
ＡＣＬ管理手段を呼出す；
コードの実行：
リソースがアクセスされるときアクセス検査へのトラップを
コード中に挿入；
変更されたコードを実行；
アクセス検査：
ＡＣＬ強制手段を呼出す；
ｉｆ アクセスが許されれば ｔｈｅｎ
継続；
ｅｌｓｅ ｉｆ アクセスが遅延されれば ｔｈｅｎ
プログラムを遅らせる；
ｅｌｓｅ ｉｆ アクセスが拒否されれば ｔｈｅｎ
プログラムを停止する；

【図９】

```
論理リソース・アクセスのチェック：
―――――――――――――――――――――
        この論理リソースの実行時パラメータ・エントリの位置を決定；
        ｗｈｉｌｅ まだ実行時パラメータ・エントリがあれば ｄｏ
            ｗｈｉｌｅ まだパラメータがあれば ｄｏ
                ｉｆ パラメータへのアクセスが許されなければ
                    ｇｏｔｏ ｇｅｔＮｅｘｔＲＰＥ；
            ｅｎｄ
            アクセス許可を戻す；

            ｇｅｔＮｅｘｔＲＰＥ： 次の実行時パラメータ
                                    ・エントリを得る；
        ｅｎｄ
        アクセス拒否を戻す；

物理リソース・アクセスのチェック：
―――――――――――――――――――――
        実行時物理リソース・テーブルの中でこの物理リソースの
        エントリの位置を決定；
        ｉｆ 最大許容量がブランクでなく、要求量＋使用量が
             最大許容量より大きければ ｔｈｅｎ
             アクセス拒否を戻す；
         予測消費率を計算；
        ｉｆ 予測消費率が最大許容消費率よりも大きければ ｔｈｅｎ
             必要な遅延を計算；
           アクセス遅延と必要な遅延を戻す；
        ｅｌｓｅ
           アクセス許可を戻す；
```

フロントページの続き

(72)発明者　ラジェヴィ・ラマスワミ
アメリカ合衆国10562、　ニューヨーク州
オッシニング　クロトン　アヴェニュー
52　アパートメント　６Ｃ

(72)発明者　ディンカール・シタラム
アメリカ合衆国10598、　ニューヨーク州
ヨークタウン　ハイツ　セス　レーン
525

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-111833
(43)Date of publication of application : 28.04.1998

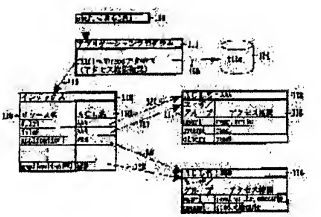
(51)Int.Cl. G06F 12/14
G06F 12/00

(21)Application number : 08-265746
(22)Date of filing : 07.10.1996

(71)Applicant : HITACHI LTD
(72)Inventor : HOSAKA TAKASHI

(54) ACCESS RIGHT MANAGEMENT SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To enable a large-scale computer system which has many resources to flexibly and easily manage rights to access by registering the relation between an access control list(ACL) and resources in a table and then performing operations for maintenance and reference.
SOLUTION: An application program 111 is started by execution 110 by a user user1. This application program 111 gains access to a file 123 named file1, but before the access, it is confirmed whether or not there is a right to gain read access to a user1's file1. For the purpose, indexes 112 are referred to and the resource file1 is retrieved. When it is found, a corresponding ACL name AAA is obtained and the entity 113 of the ACL is obtained from it. The user user1 is retrieved from it to know that the user1 has a read right among its own access rights. Here, the read access is actually gained.



(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－111833

(43)公開日　平成10年(1998)４月28日

(51)Int.Cl.⁶		識別記号	ＦＩ		
Ｇ０６Ｆ	12/14	３１０	Ｇ０６Ｆ	12/14	３１０Ｋ
	12/00	５３７		12/00	５３７Ａ

審査請求　未請求　請求項の数５　ＯＬ　（全５頁）

(21)出願番号　特願平8－265746

(22)出願日　平成８年(1996)10月７日

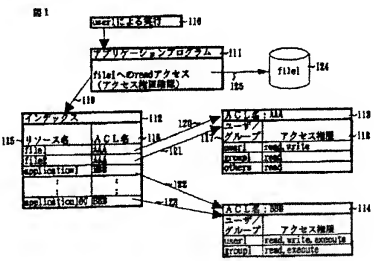
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　保坂　崇
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　アクセス権限管理方式

(57)【要約】
【課題】大規模で大量のリソースを有する計算機システムにおいてアクセス権限を柔軟な形で容易に管理することを可能にする。
【解決手段】ファイル名ｆｉｌｅ１のファイル１２４にアクセスするアプリケーション１１１がアクセス権限を確認するためにインデックステーブル１１２を参照してｆｉｌｅ１に対応するＡＣＬを検索しＡＡＡというＡＣＬ１１３得る。ここで実際にアクセス権限の有無を確認する。インデックステーブル１１２上では複数のリソースを１つのＡＣＬにまとめて関連づけされている様子を示している。



【特許請求の範囲】
【請求項１】ユーザファイル、ユーザアプリケーション、コマンド等の大量のリソースを有する計算機システムにおけるアクセス権限管理の際に任意の複数リソースをグルーピングして管理することでアクセス権限の管理の容易化を計ることができるアクセス権限管理方式。
【請求項２】インデックスを利用することでアクセス主体のアクセス権限をリストとして管理しているアクセスコントロールリスト(ＡＣＬ)とリソースとを結び付け、１つのＡＣＬで複数のリソースのアクセス権限管理を行う請求項１記載のアクセス権限管理方式。
【請求項３】アクセス権限の検証を行うためリソースとＡＣＬの対応付けを行いアクセスの主体のアクセス権限の有無を確認する請求項１記載のアクセス権限管理方式。
【請求項４】アクセス権限の登録・更新を行うためリソースとＡＣＬの対応付けを行い、複数のリソースのアクセス権限を一括して操作する請求項１記載のアクセス権限管理方式。
【請求項５】新規リソース追加の際に既存リソースと同一のＡＣＬを用いて権限管理を行う場合、新たなＡＣＬを作成せずに既存リソースのＡＣＬと結びつける操作を提供する請求項１記載のアクセス権限管理方式。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、大規模な計算機システムにおけるアクセス権限の管理方式に関わるものである。
【０００２】
【従来の技術】従来のアクセス権限管理方式ではアクセスされる各々の対象物に対して、アクセスする主体とアクセス権限を組にしたリストを作成し、これをアクセスコントロールリスト（ＡＣＬ）として実際にアクセスが行われる時にそのアクセス方法に該当するアクセス権限を有するか否かをＡＣＬを参照することにより制限できた。しかし各々の対象物毎にこのＡＣＬを作成・管理するために大量のリソースを有する大規模な計算機システムにおいては管理が難しいという従来の問題点が残っている。この問題を解決するために「セキュリティ管理の技術、日科技連、ｐ．２３～２４」に記載されている名称規則による総称指定、グループ定義といった手段が存在していたがそれぞれ名称が一部固定されてしまう、アクセスの対象物のアクセス権限管理としては変わらず管理に労力を要するといった問題が残されている。
【０００３】
【発明が解決しようとする課題】上記の従来技術では、大規模で大量のリソースを有する計算機システムにおいてアクセス権限の管理を柔軟な形で容易にすることが望まれていた。
【０００４】本発明の目的はこのアクセス権限の管理を柔軟な形で容易にする手段を提供することにある。
【０００５】
【課題を解決するための手段】上記の目的はアクセスコントロールリスト（ＡＣＬ）とリソースの関連をテーブルに登録し、この上で保守、参照といった操作を行う手段を提供し、また、必要により任意の複数リソースからなるグループを作成し、これからある１つのＡＣＬを参照する手段を提供することによって達成される。
【０００６】
【発明の実施の形態】以下、本発明の一実施例について図面で説明する。
【０００７】図１は本発明によるアクセス権限管理方式を用いた場合のアプリケーションによるアクセス権限の確認方式を示す図である。ユーザｕｓｅｒ１による実行１１０によりアプリケーションプログラム１１１が起動される。アプリケーションプログラム１１１ではｆｉｌｅ１というファイル１２４へのｒｅａｄアクセスがある。このファイルへのｒｅａｄアクセスの前にｕｓｅｒ１のｆｉｌｅ１に対するｒｅａｄアクセス権の有無確認を行う。そのためにまず、インデックス１１２を参照する。ここでリソース名ｆｉｌｅ１を検索する。見つかったら対応するＡＣＬ名ＡＡＡを取得する。この得られてＡＣＬ名ＡＡＡからＡＣＬの実体１１３を得る。この中でユーザｕｓｅｒ１を検索しｕｓｅｒ１の持つアクセス権限にｒｅａｄ権のあることが分かる。ここでｕｓｅｒ１によって起動されたアプリケーションプログラム１１１にｆｉｌｅ１へのｒｅａｄアクセスが許され、実際にｆｉｌｅ１に対してｒｅａｄアクセスが行われる。
【０００８】図２は本発明によるアクセス権限管理方式におけるＡＣＬ操作コマンドの操作を示す図である。ＡＣＬ操作コマンド２１０はアクセス権限の確認に用いるインデックス２１１とＡＣＬの実体２１２、２１３の操作する手段を提供している。インデックスの操作としてリソース名２１４の登録、リソース名２１４とＡＣＬ名２１５との関連づけを行う。ＡＣＬの操作としてはＡＣＬの作成、ユーザ／グループ２１６の追加・削除、アクセス権限２１７の変更等がある。
【０００９】これを用いることでＡＣＬによる権限の管理を行う。
【００１０】図３は図１におけるリソースとＡＣＬの関連づけの概念を示す図である。リソース群に登録したｆｉｌｅ１からｆｉｌｅ３は同じＡＣＬ：ＡＡＡと関連づけられている。これでＡＣＬ：ＡＡＡを操作することでｆｉｌｅ１からｆｉｌｅ３のＡＣＬを一括して操作することが可能になる。同様にａｐｐｌｉｃａｔｉｏｎ１からａｐｐｌｉｃａｔｉｏｎ１００はＡＣＬ：ＢＢＢに関連づけられている。従ってＡＣＬ：ＢＢＢを操作することでａｐｐｌｉｃａｔｉｏｎ１からａｐｐｌｉｃａｔｉｏｎ１００のＡＣＬを一括して操作できることになる。ｆｉｌｅ１０とａｐｐｌｉｃａｔｉｏｎ２００はＡＣ

Ｌ：ＣＣＣと関連づけられている。このように各ＡＣＬに関連づけられるリソース名はユーザの任意の名称での登録が可能となっている。また任意の組み合わせも可能となっている。

【００１１】図４は図２におけるＡＣＬ操作コマンドの操作例を示す図である。４１１はユーザのアクセス権限登録の例を示す。ＡＣＬ操作コマンドにパラメタとしてユーザ名ｕｓｅｒ１、ｕｓｅｒ１に与える権限としてｒｅａｄ、そしてＡＣＬ名ＡＡＡを指定する。この操作によりＡＣＬ：ＡＡＡにｕｓｅｒ１のｒｅａｄ権限が登録される。４１２にＡＣＬに対するリソースの関連づけ例を示す。

【００１２】ＡＣＬ操作コマンドのパラメタとしてＡＣＬ名、リソース名の列を指定する。

【００１３】この操作によりＡＣＬ：ＡＡＡとリソース１、リソース２、リソース３とが関連づけされる。

【００１４】図５は図１におけるアクセス権限検証のフローチャートである。ｕｓｅｒ１によりｆｉｌｅ１に対するｒｅａｄ操作が開始する（５１０）。まずインデックスよりリソース名ｆｉｌｅ１を検索する（５１１）。対応するＡＣＬ名ＡＡＡを取得する（５１２）。ＡＣＬ　ＡＡＡを開く（５１３）。ユーザｕｓｅｒ１をＡＣＬ上で検索する（５１４）。ｕｓｅｒ１が見つかったらｒｅａｄ権限の有無を確認する（５１５）。もし有れば実際にファイルアクセスを実行（５１７）してｆｉｌｅ１へのｒｅａｄ処理操作を終了する（５１８）。もしｕｓｅｒ１にｒｅａｄ権限がなければファイルアクセスエラー発生（５１６）としてｆｉｌｅ１へのｒｅａｄ処理操作を終了する（５１８）。

【００１５】

【発明の効果】以上説明したように、本発明によれば、大量のリソースを有する大規模な計算機システムにおいて柔軟で且つ容易にアクセス権限の管理を行うことを可能にする。

【図面の簡単な説明】

【図１】本発明によるアクセス権限管理方式を用いた場合のアプリケーションによるアクセス権限の確認方式を示す図である。

【図２】本発明によるアクセス権限管理方式におけるＡＣＬ操作コマンドの操作を示す図である。

【図３】図１におけるリソースとＡＣＬの関連づけされた様子を示す図である。

【図４】図２におけるＡＣＬ操作コマンドの操作例を示す図である。

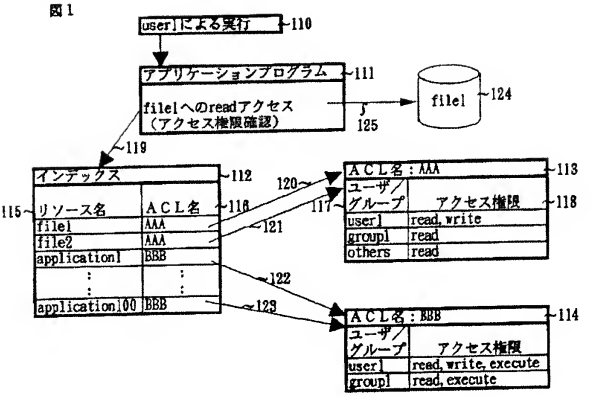
【図５】図１におけるアクセス権限検証のフローチャートである。

【符号の説明】

１０…アプリケーションプログラム、１１…リソース、１２…インデックステーブル、１３…ＡＣＬ（ＡＣＬ名：ＡＣＬ１）、１４…ＡＣＬ（ＡＣＬ名：ＡＣＬ２）、１５…リソース名欄、１６…ＡＣＬ名欄、１７…ユーザ／グループ欄、１８…アクセス権限欄１９…インデックス上のＡＣＬ名：ＡＣＬ１とＡＣＬの実体の関連づけ、２０…インデックス上のＡＣＬ名：ＡＣＬ２とＡＣＬの実体の関連づけ、２１…アプリケーションプログラムからアクセス権限確認、２２…アプリケーションプログラムからリソースへのアクセス、２３…ＡＣＬ操作コマンドによるインデックスの操作、２４…ＡＣＬ操作コマンドによるＡＣＬ１の操作、２５…ＡＣＬ操作コマンドによるＡＣＬ２の操作、１１０…ｕｓｅｒ１による実行操作、１１１…アプリケーションプログラ、１１２…インデックステーブル、１１３…ＡＣＬ（ＡＣＬ名：ＡＡＡ）、１１４…ＡＣＬ（ＡＣＬ名：ＢＢＢ）、１１５…リソース名欄、１１６…ＡＣＬ名欄、１１７…ユーザ／グループ欄、１１８…アクセス権限欄、１１９…アプリケーションプログラムからアクセス権限確認、１２０…インデックス上のＡＣＬ名：ＡＡＡとＡＣＬの実体の関連づけ、１２１…インデックス上のＡＣＬ名：ＡＡＡとＡＣＬの実体の関連づけ、１２２…インデックス上のＡＣＬ名：ＢＢＢとＡＣＬの実体の関連づけ、１２３…インデックス上のＡＣＬ名：ＢＢＢとＡＣＬの実体の関連づけ、１２４…ファイル（ファイル名：ｆｉｌｅ１）、１２５…アプリケーションプログラムからファイルへのアクセス、２１０…ＡＣＬ操作コマンド、２１１…インデックステーブル、２１２…ＡＣＬ（ＡＣＬ名：ＡＡＡ）、２１３…ＡＣＬ（ＡＣＬ名：ＢＢＢ）、２１４…リソース名欄、２１５…ＡＣＬ名欄、２１６…ユーザ／グループ欄、２１７…アクセス権限欄、２１８…ＡＣＬ操作コマンドによるインデックステーブルに対する操作、２１９…ＡＣＬ操作コマンドによるＡＣＬ　ＡＡＡに対する操作、２２０…ＡＣＬ操作コマンドによるＡＣＬ　ＢＢＢに対する操作、３１１…リソース群、３１２…ＡＣＬ群、３１３…リソース　ｆｉｌｅ１、３１４…リソース　ｆｉｌｅ２、３１５…リソース　ｆｉｌｅ３、３１６…ＡＣＬ　ＡＡＡ、４１１…ユーザのアクセス権限登録例、４１２…ＡＣＬに対するリソースの関連付け例

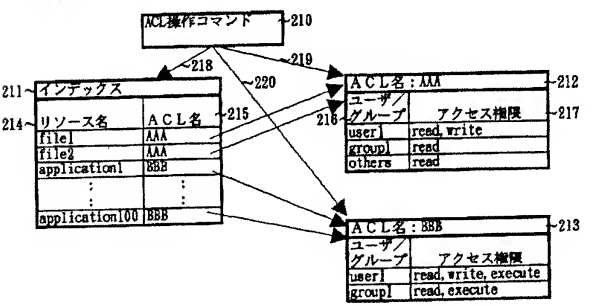
【図１】

図１



【図２】

図２



【図４】

図４

ユーザのアクセス権限登録例

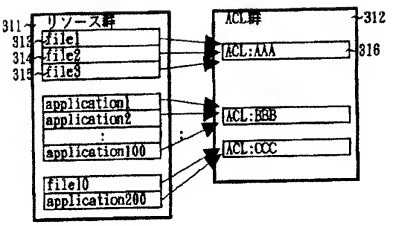
411　ACL操作コマンド　user1:read　AAA

ACLに対するリソースの関連付け例

412　ACL操作コマンド　AAA　リソース1　リソース2　リソース3

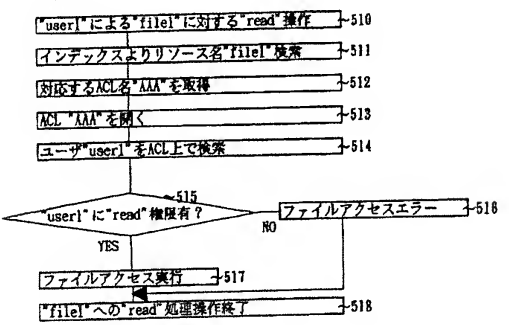
【図３】

図３


311
リソース群
313
file1
314
file2
315
file3
application1
application2
application100
file10
application200
ACL群
312
ACL:AAA
316
ACL:BBB
ACL:CCC

【図５】

図５


"user1"による"file1"に対する"read"操作
510
インデックスよりリソース名"file1"検索
511
対応するACL名"AAA"を取得
512
ACL "AAA"を開く
513
ユーザ"user1"をACL上で検索
514
515
"user1"に"read"権限有？
NO
ファイルアクセスエラー
516
YES
ファイルアクセス実行
517
"file1"への"read"処理操作終了
518